Software Architecture and Design

Documentation I

Nguyen Duc Man, Msc
DTU International School
Email: mannd@duytan.edu.vn
Cell: 0904 235 945

Lecture Topics

- **Introduce principles and techniques of sound architectural documentation**
  - General principles
  - Perspectives and viewtypes
  - Example viewtypes

Review: Guidance For the Architect*

1) Establish perspective and set context
2) Select a perspective, and begin decomposition
3) Switch perspective as necessary and continue decomposition and refinement
4) Document as you design
5) Evaluate the architecture
6) Iterate as necessary

Example

Is this good software architecture documentation?

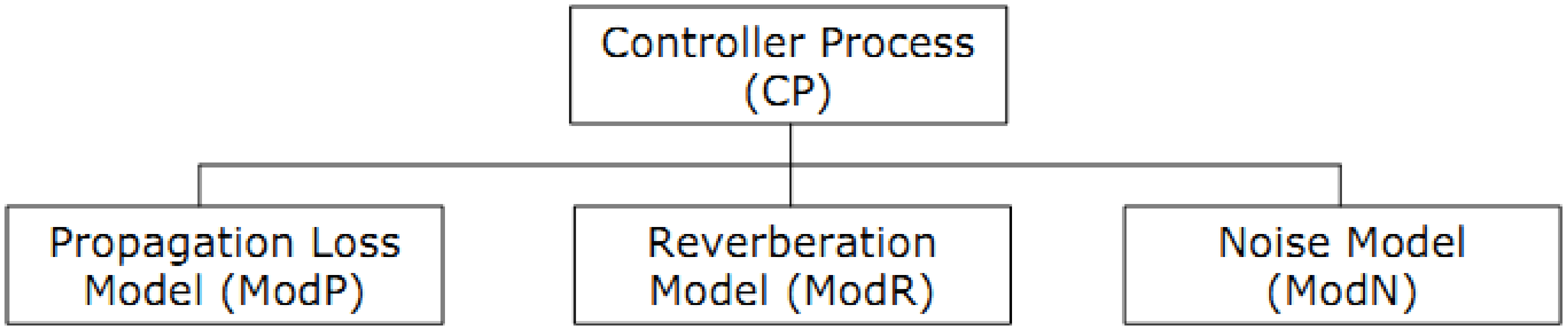
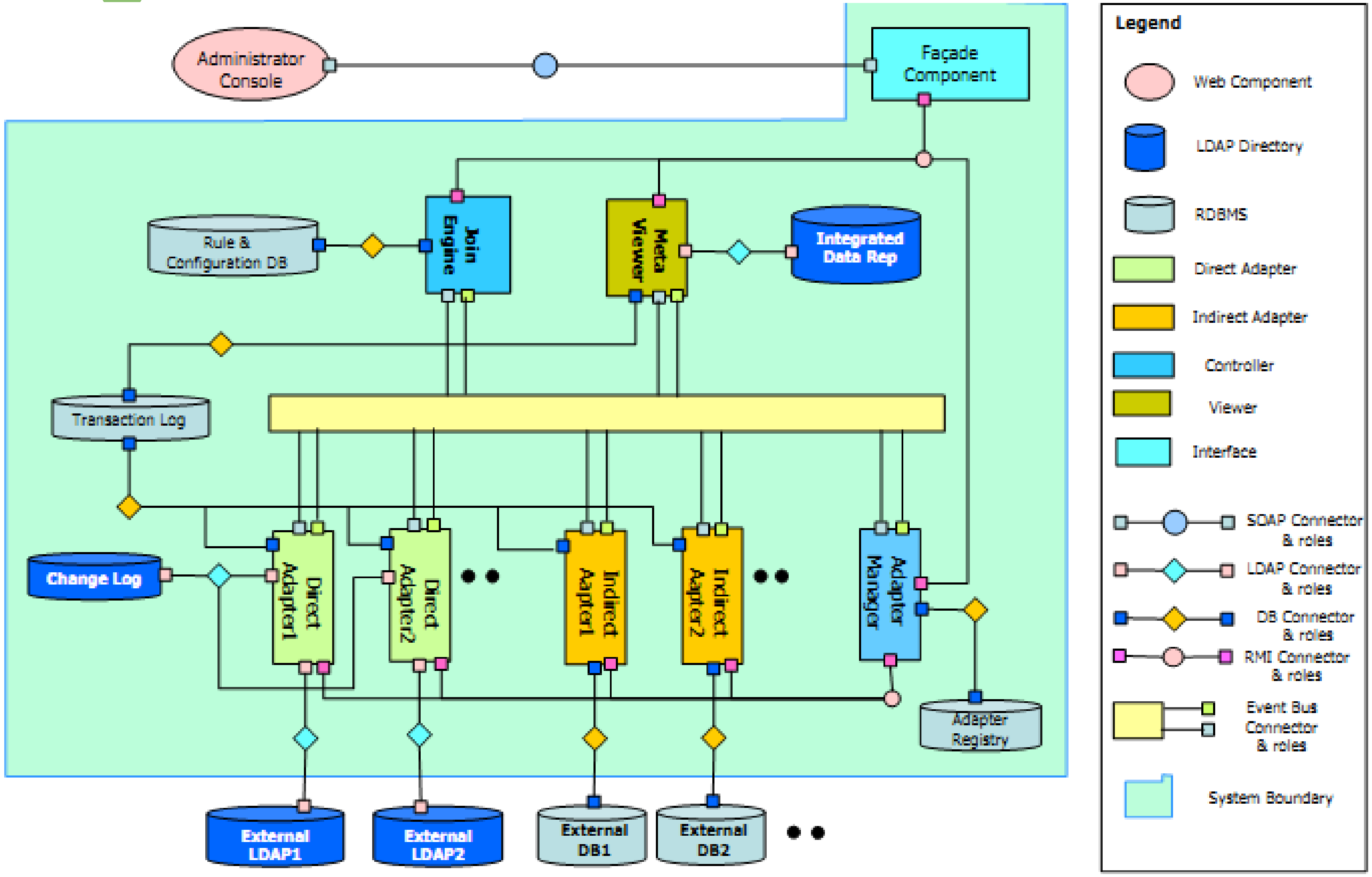


Figure X: Overall Software System Structure

What’s Wrong?

- **We agreed it was not because too many things are left unspecified:**
  - What kind of components?
  - What kind of connectors?
  - What do the circles mean?
  - What is the significance of the layout?
  - Why is control process on a higher level?
- **Effective architecture descriptions require three fundamental things:**
  - Drawings, a legend, and prose

Is This Any Better?



Much Better…

- **This picture addresses a few questions:**
  - We know there is a difference between the elements
  - We know there is a difference between the relations
  - We have a better idea of overall structure of the system

Still Not Complete – 1

- **While this is a meaningful picture, many more questions remain:**
  - What perspective is this?
  - How does this map to hardware?
  - What code modules make up the various parts of the system?
  - Where does the user interact with the system?
  - What is the dynamic behavior of the system?
  - Rationale: why this design?

Still Not Complete – 2

- **Pictures alone do not suffice for architecture documentation**
  - Pictures can be interpreted differently by different readers
  - Even "formal notation" is open to interpretation
  - Formal notation is not understood by all readers
- **Pictures require supporting documentation**
  - A "view" of a system consists of a picture AND supporting documentation
- **One picture or "view" alone can't tell the whole story about the system**

What I See In Practice

- **What documentation?**
- **In practice, software architecture documentation today includes:**
  - UML
  - box and line drawings
- **Tools most often used to create architectural documentation**
  - Visio
  - PowerPoint
  - Word
  - Rational Rose

The Problem

- **Architecture documentation is for communicating complex information and ideas:**
  - If you can’t explain it to someone, it has little value
  - Poor documentation is often a symptom of sloppy and incomplete thinking
- **In practice today’s documentation consists of:**
  - Ambiguous box-and-line diagrams
  - Poor justification of rationale
  - No discussion of alternatives
  - Inconsistent use of notations
  - Confusing combinations of view types
  - In consistent perspectives

Importance of Documentation

- **Architecture documentation is important if and only if communication of the architecture is important**
  - How can an architecture be used if it cannot be understood?
  - How can it be understood if it cannot be communicated?
  - Architectural documentation must be descriptive and prescriptive
- **Documentation speaks for the architect, today and for the lifetime of the system**

Seven Principles

1. Write from the point of view of the stakeholder
2. Avoid unnecessary repetition
3. Avoid ambiguity
4. Use a standard organization
5. Record rationale
6. Keep documentation current but not too current
7. Review documentation for fitness of purpose

Stakeholders Point Of View

- **What will the reader want to know when reading a document?**
  - Make information easy to find!
  - Your reader will appreciate your effort and be more likely to read your document
- **Avoid writing for your (i.e. the writer's) convenience**
  - stream of consciousness: the order is that in which things occurred to the writer
  - stream of execution: the order is that in which things occur in the computer during program or task execution

Avoid Unnecessary Repetition

- **Each kind of information should be recorded in exactly one place**
  - This makes documents easier to use and easier to change and more likely that they will be maintained over the lifetime of the system
  - Repetition often confuses, because the information is repeated in slightly different ways

Avoid Ambiguity

- **Architecture documentation is a communications vehicle. If the reader misunderstands, the documentation has failed**
- **Box-and-line diagrams are a common form of architectural notation, but what do they mean?**
  - Always include a key or legend
  - If a common language and/or notation is used, point to the formal definition – don't assume everyone knows the notation
  - Give the meaning of each symbol and each line
  - Remain consistent in the use of the symbols

Use a Standard Organization

- **Establish it, make sure your writers adhere to it, and make sure that readers know what it is**
  - helps the reader navigate and find information
  - helps the writer place information and measure work left to be done
  - embodies completeness rules, and helps writers check for completeness as the write
- **Organize the documentation for the reader's ease of reference not for the convenience of the writer**
  - A successful document will be referred to many times

Record Rationale

- **Why did you make certain design decisions?**
  - Next week, next year, or next decade, how will you remember? How will the next designer know?
  - Recording rationale requires a cultivated discipline, but saves enormous time in the long run.
  - Record rejected alternatives and reasons for rejection as well.
  - Make it a habit to carry an engineering notebook with you throughout the project.

Keep Documentation Current...

- Documentation that is incomplete, out of date, does not reflect truth
- Documentation that is kept current is used
- With current documentation, questions are most efficiently answered by referring the questioner to the documentation
- If a question cannot be answered with a document, fix the document and then refer the questioner to it
- This sends a powerful message to readers

...But Not Too Current

- During the design process, decisions are considered and re-considered at high frequency
- Revising the documentation every five minutes will result in unnecessary expense, inertia, and resistance to change because its costly to change documentation
- Choose points in the development plan when documentation is brought up to date – schedule it like you would any other task
- Follow a release strategy that makes sense for your project

Review For Fitness of Purpose

- **Only the intended users of a document can tell you if it**
  - contains the right information
  - presents the information in a useful way
  - satisfies their needs
- **Plan to review your documents with stakeholders for whom it was created**

Architectural Views – 1

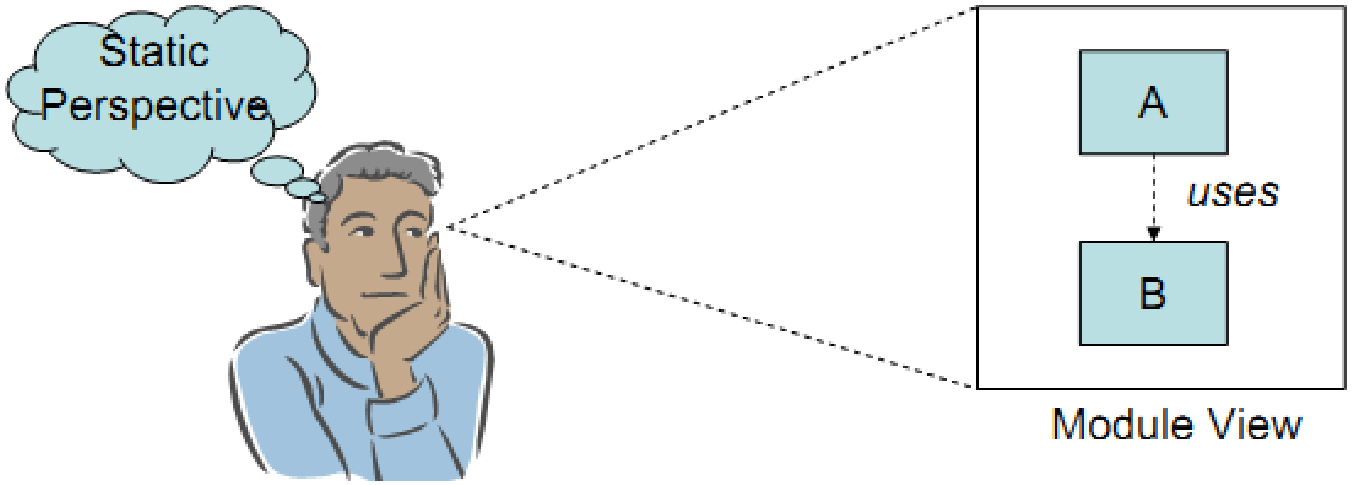
- **Architecture design can be very complex and its almost always too complicated to be seen all at once**
- **Software intensive systems have many structures or views**
  - Just as buildings have drawings describing electrical systems, plumbing, structure, so it is with software
  - No single representation structure or artifact can be the architecture
  - The set of candidate structures is not fixed or prescribed: architects need to select what is useful for analysis or communication

Architectural Views – 2

- **Systems are composed of many structures**
  - modules, showing composition /decomposition, mapping to code units
  - processes and how they synchronize and exchange information
  - applications and the mechanisms they use to interact with other applications
  - how software is deployed on hardware
  - how teams cooperate to build the system
  - …and many others

Architectural Views – 3

- A view is a partial representation of some set of system elements and the relations between them
  - Not all system elements – some of them
- A view binds elements and relations from a particular perspective



Recall: Perspective

- **Structures are real things and software intensive systems have many different kinds of structures: code, processes, hardware, etc.**
- **The structures we see or can reason about depend upon perspective**
  - The structures we document or analyze in the implementation depend upon the perspective
- **Perspective is an intellectual construct –if you don’t get right in your head, you won’t get it right on paper!**

From Perspective to View

- **An architect must consider the software from at least three perspectives:**
  - How is it structured as a set of code elements?
  - How is it structured as a set of elements that have run-time behavior and interactions?
  - How does it relate to non-software elements in its environment?
- **Documenting the design from each of the perspectives yields one of three types of views which we call view types**

Views – 1

- **A view is a documentation construct**
  - A view is a representation of a set of system elements and the relations associated with them from a particular perspective
  - A view does not represent the whole system design in one picture. A view shows part of the system, some of the elements of the system
  - A view binds a set of elements and a set of relationships
  - The elements and relationships that are permissible in a particular view (view type) depend upon the perspective

Views – 2

- **Structures documented from the static perspective:**
  - Module viewtype - shows elements that are units of implementation (static perspective)
- **Structures documented from the dynamic perspective:**
  - Component-and-connector (C&C) viewtype -shows elements that have run-time behavior and interaction
- **Structures documented from the physical perspective:**
  - Allocation viewtype - how software structures are allocated to non-software structures

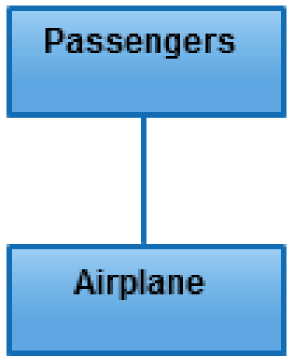
Perspective, Structures, Relationships, and Views

In this session, we will focus on views…

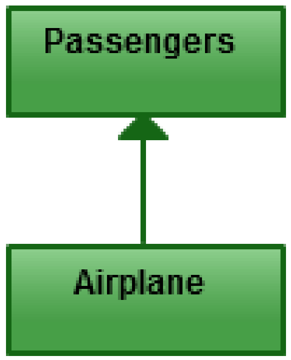
Perspective	Example Structures	Example Relationships	Views
Dynamic	Processes Threads :	Dataflow Events :	Component and Connector
Static	Layers Code Modules :	Depends Calls :	Module
Physical	Computers Sensors :	Serial Line Wireless :	Allocation

Module Views

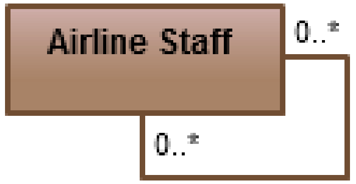
- **Perspective: Static**
- **Elements: Modules. A module is a code unit that implements a set of responsibilities**
- **Relations: Relations among modules include:**
  - A is part of B. This defines a part-whole relation among modules
  - A depends on B. This defines a dependency relation among modules
  - A is a B. This defines specialization and generalization relations among modules
- **Properties: name, responsibilities, visibility, interface**



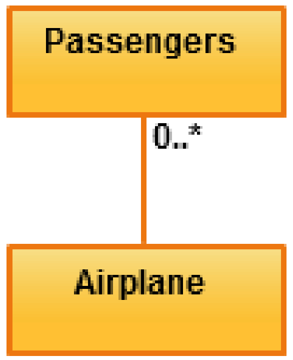
Association



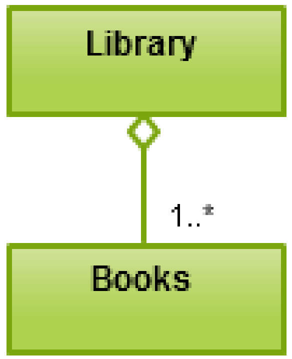
Directed Asscoation



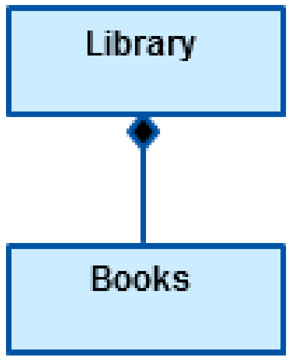
Reflexive Assciation



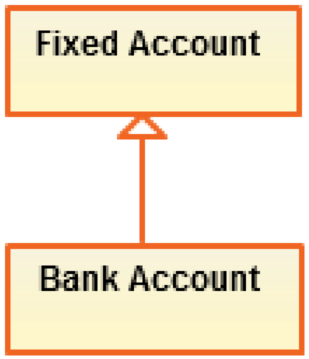
Multiplicity



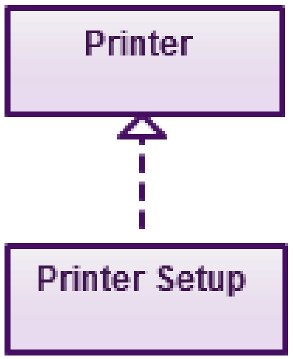
Aggregation



Composition

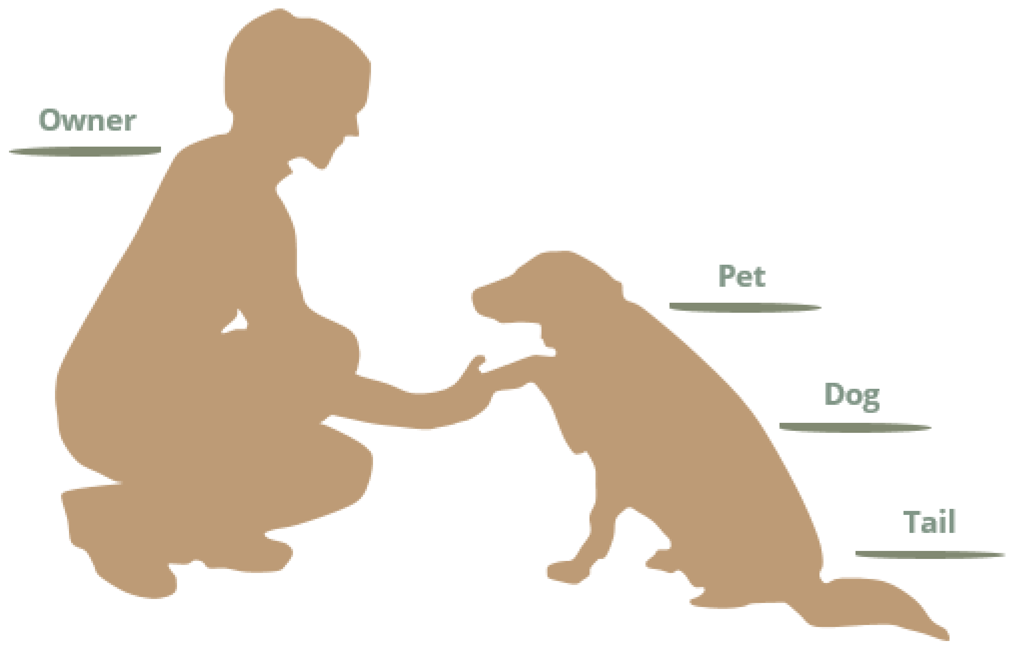


Inheritance



Realization

Association • Aggregation • Composition



We see the following relationships:
•owners feed pets, pets please owners (**association**)
•a tail is a part of both dogs and cats (**aggregation** / **composition**)
•a cat is a kind of pet (**inheritance** / **generalization**)

Component-and-Connector Views

- Perspective: Dynamic
- Elements:
  - Components: principal units of run-time interaction and data stores
  - Connectors: interaction mechanisms
- Relations: Attachments of components' to connectors'
- Properties:
  - name
  - dynamic functional responsibilities
  - quality attribute volumetrics: how much, how fast, how many, how often,…

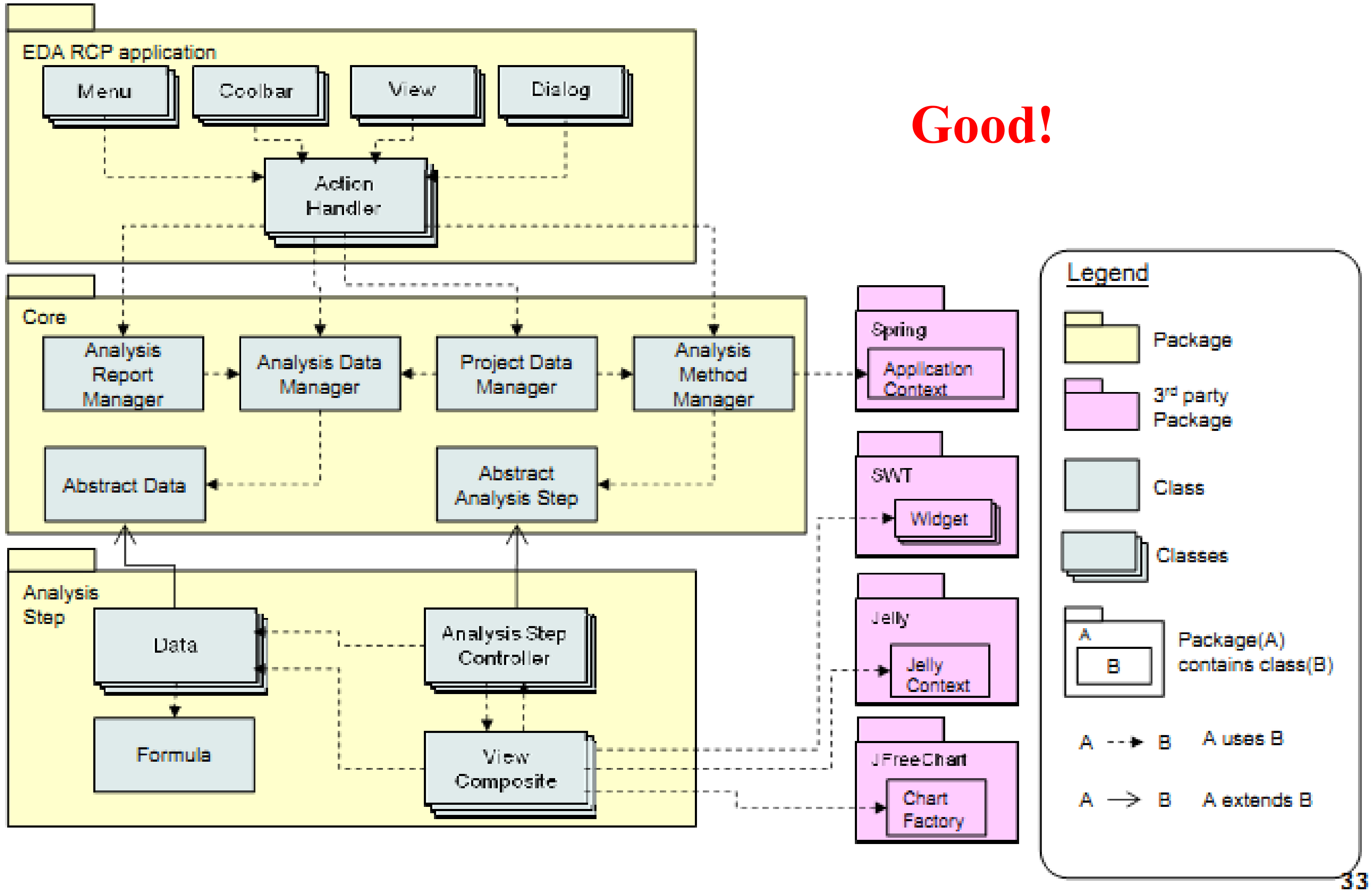
Allocation Views

- Perspective: Physical
- Elements:
  - software elements usually as defined in module or C&C viewtypes
  - physical elements from the operational environment
- Relations: varies, but often includes "allocated to", "connected to"
- Properties: various, according to physical elements and/or what is being related

Examples

- **Now lets take a look at various kinds of view types:**
  - module viewtypes
  - C&C viewtypes
  - allocation viewtypes
- **For each viewtype, we will look at a good example and a poor example and critique each**

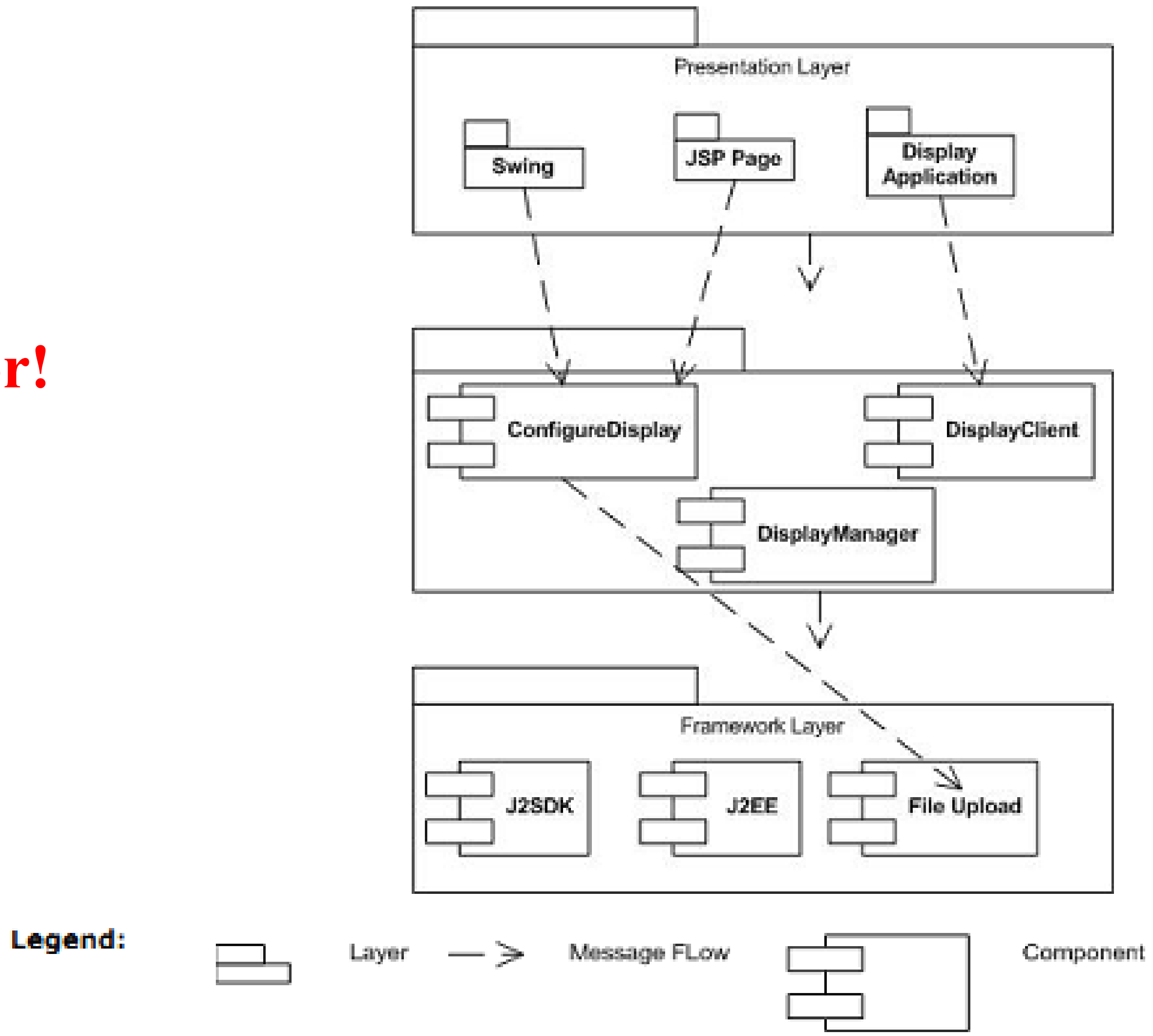
Module Viewtype



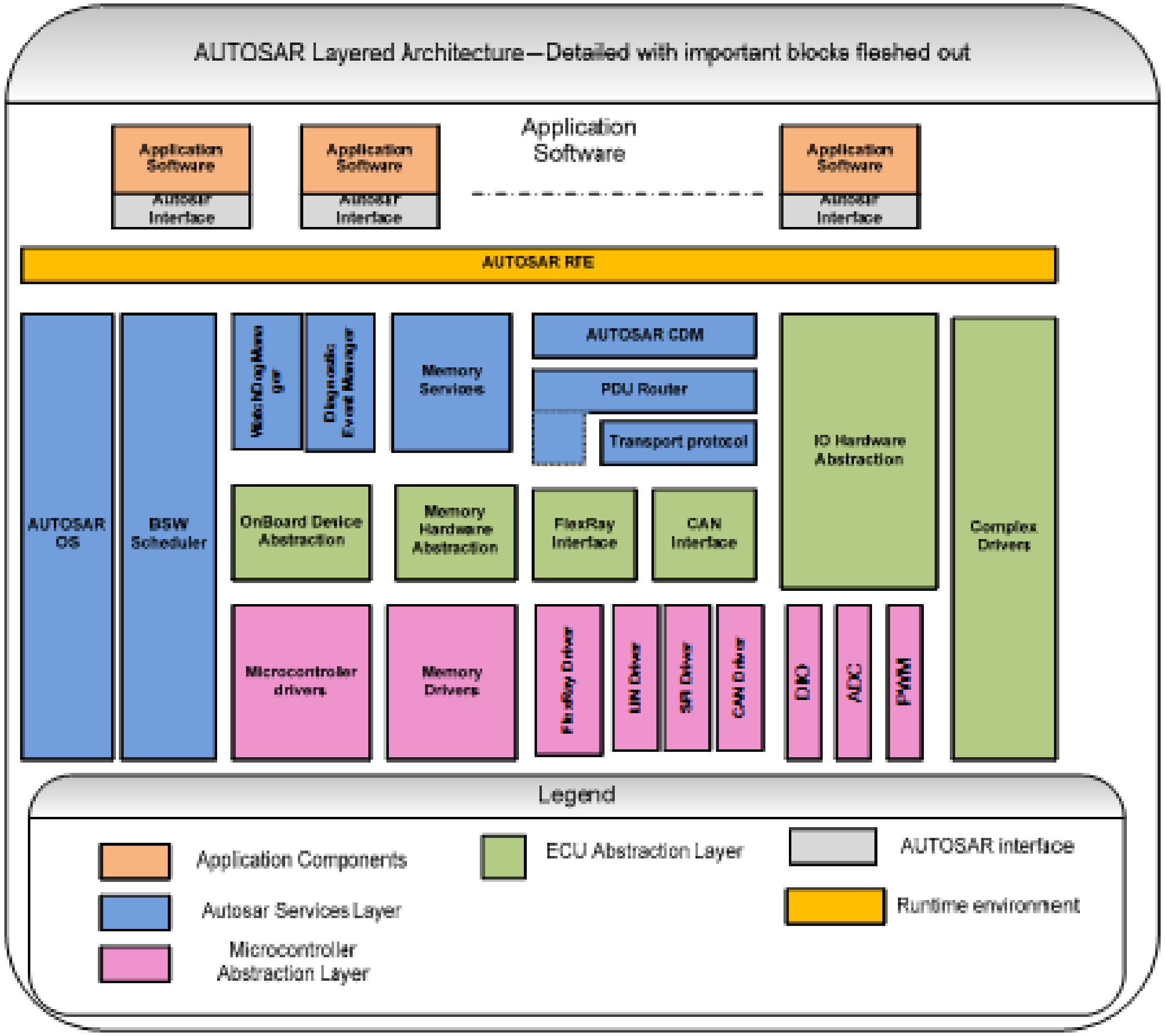
Good!

Module Viewtype

Poor!



Module Viewtype

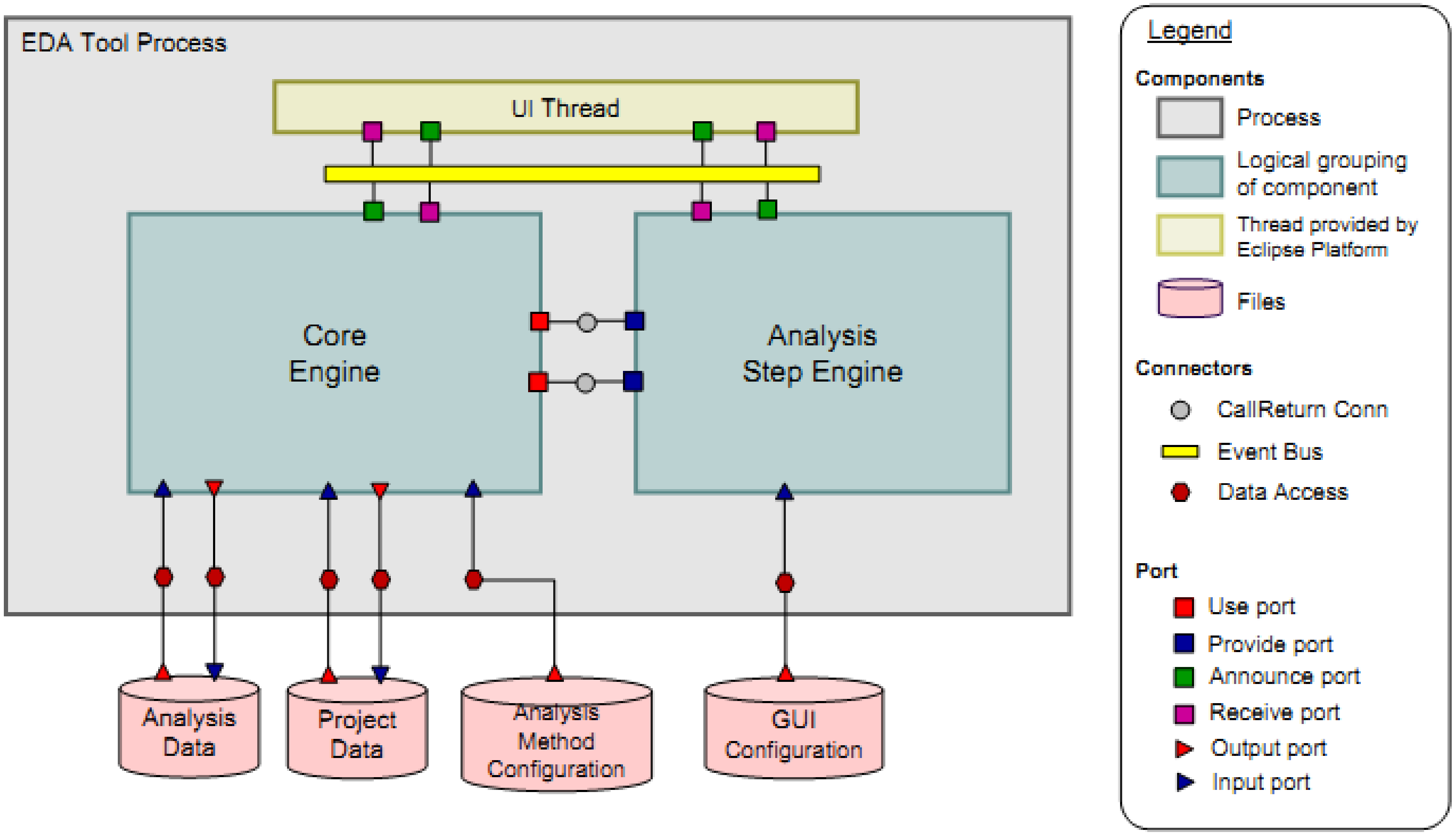


Poor!

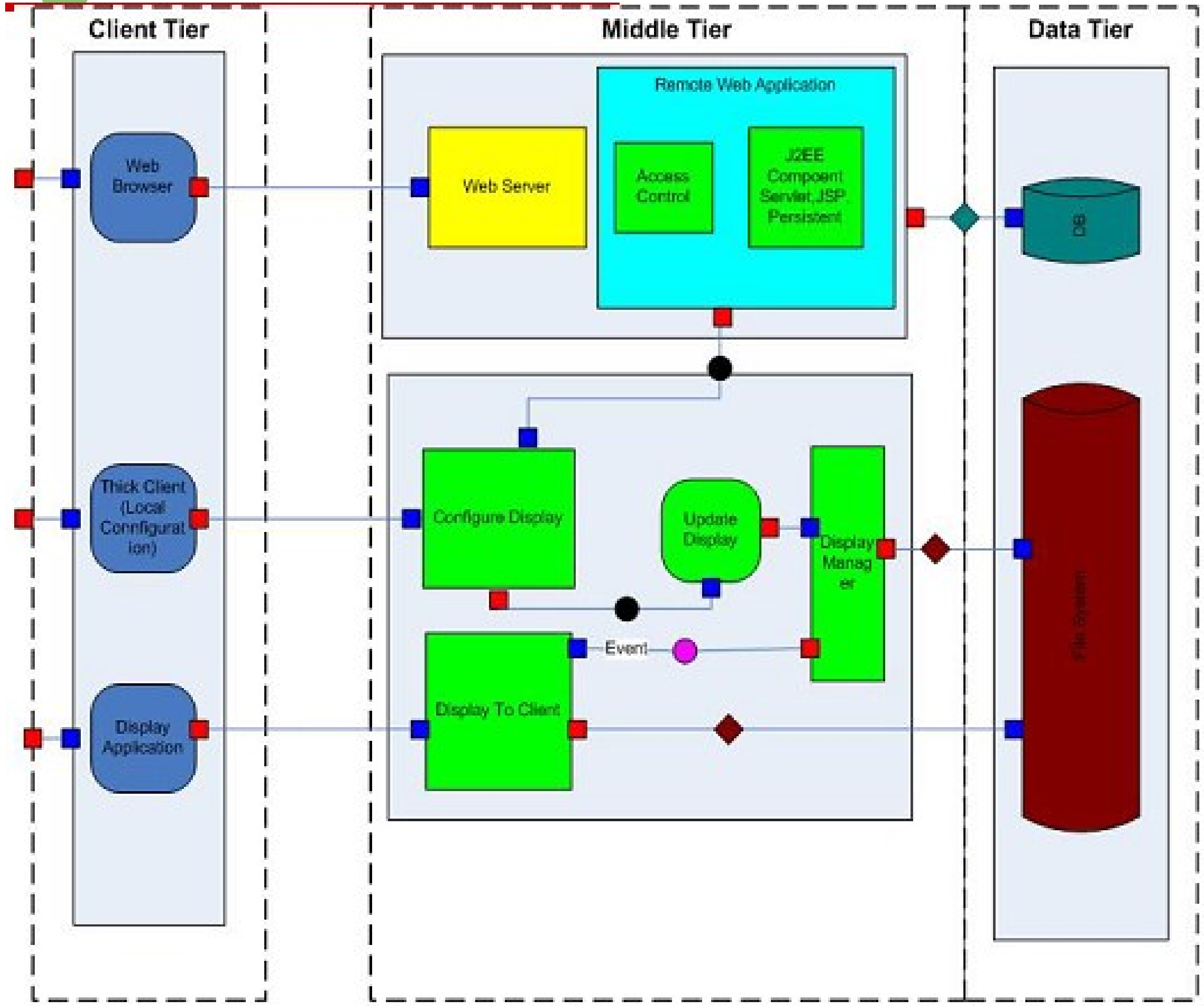
Common Errors

- **Too much implementation detail**
  - Class diagrams are often too low level
  - Need to understand important groupings
- **Failure to include relevant libraries**
- **Confusion in layered views**
  - e.g., Relations within a layer not specified
  - e.g., Not clear what are the visibility restrictions
  - e.g., Overuse of “sidecars”
- **Mixed perspectives**

C&C Viewtype (Good)



C&C Viewtype (Poor)

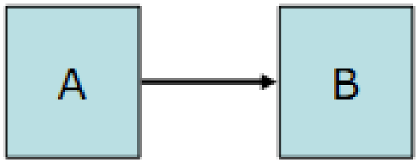


Common Errors – 1

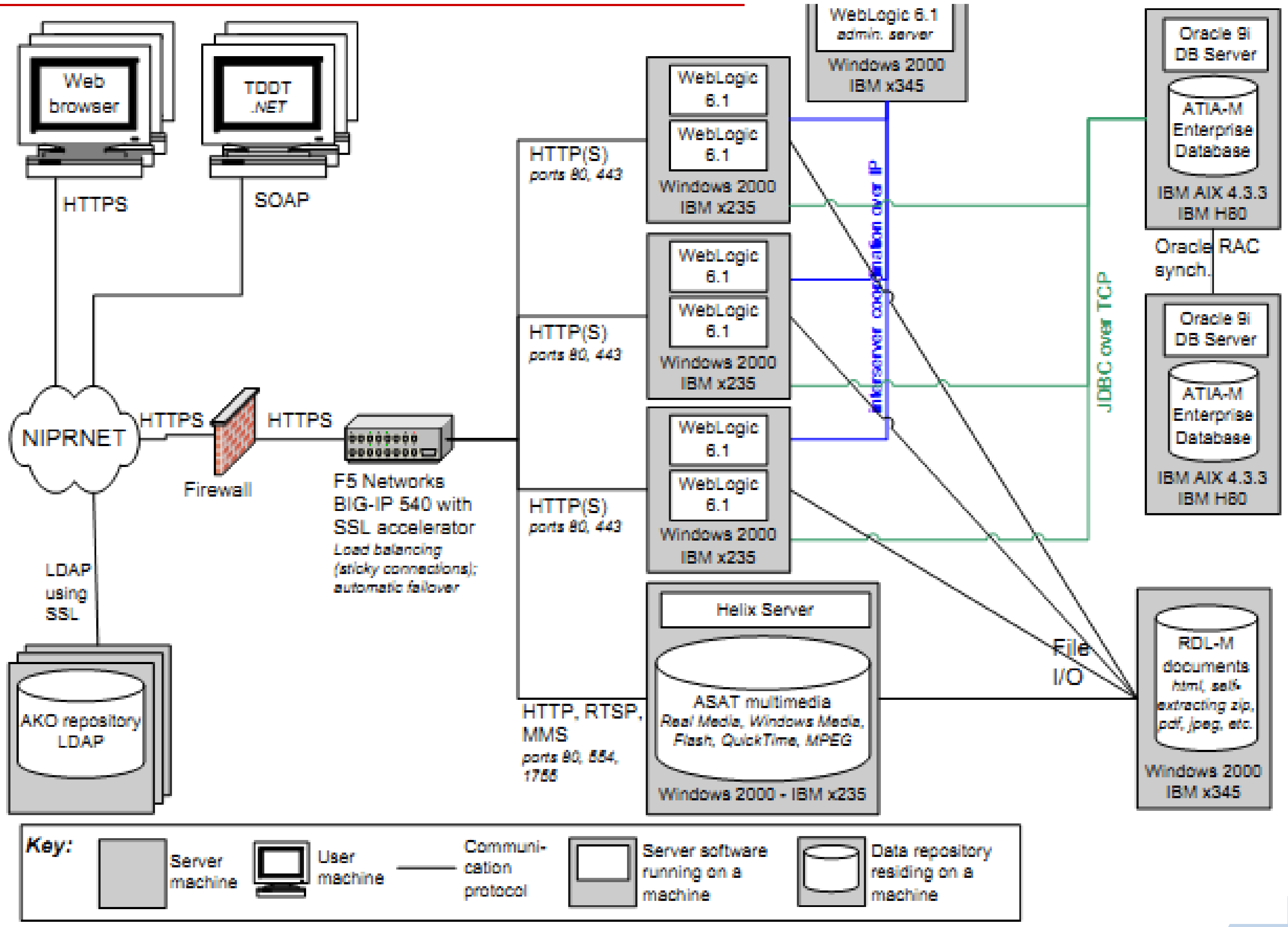
- Mixed perspectives especially between static and dynamic perspectives
- Failure to distinguish between different kinds of elements
  - elements – what kind?
  - components – what kind?
  - connectors – what kind?
- Missing connectors – “floating” elements not connected to anything

Common Errors – 2

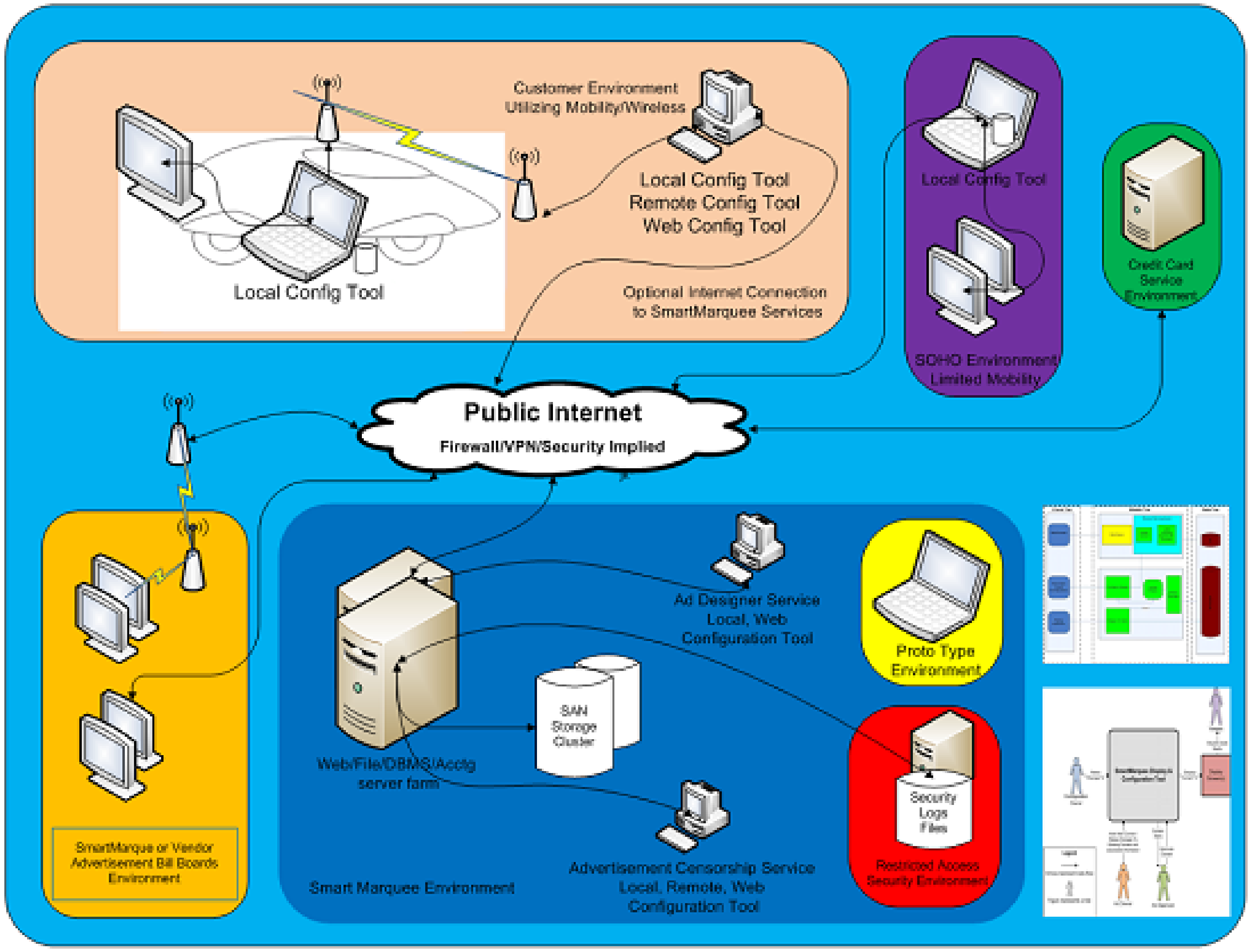
- Unclear use of arrows – for example, does this drawing mean:
  - A passes control to B?
  - A signals B?
  - A gets a value from B?
  - A streams data to B?
  - A sends a message or event to B?
  - A calls B?



Allocation Viewtype (Good)



Allocation Viewtype (Poor)



Common Errors

- Very imprecise
- Mixes hardware with all other perspectives
- Failure to show how software elements (especially those from the dynamic perspective) map to hardware
- Poor scoping: too little and too much detail

Session Summary

- **Specifically we discussed:**
  - 7 points for creating sound architectural documentation
  - architectural views, perspectives and viewtype
  - module, C&C, and allocation viewtypes and critiqued some examples

- Lattanze, A. *Architecting Software Intensive Systems.* New York, NY: Auerbach, 2008
- Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003
- Clements P.; Bachmann F.; Bass L.; Garlan G.; Ivers J.; Little R.; Nord R.; Stafford J.; *Documenting Software Architectures: Views and Beyond,* Reading, MA: Addison-Wesley, 2002

Homework

- **Mỗi nhóm đọc và tóm tắt Section 2, sách Architecting.Software.Intensive.Systems.A.Practitioners.Guide.Nov.2008.**
  - Đọc và tóm tắt từ p149 – 358
  - Mỗi chương tóm tắt từ 1.5-2 trang A4, bằng tiếng Việt
- **Làm Assignment 3**

Q&A